UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
10/646,289	08/21/2003	Son Ho	MP0390.1	9390

26703 7590 04/29/2008
HARNESS, DICKEY & PIERCE P.L.C.
5445 CORPORATE DRIVE
SUITE 200
TROY, MI 48098

EXAMINER	
PATEL, KAUSHIKKUMAR M	

ART UNIT	PAPER NUMBER
2188	

MAIL DATE	DELIVERY MODE
04/29/2008	PAPER

Please find below and/or attached an Office communication concerning this application or proceeding.

The time period for reply, if any, is set in the attached communication.

UNITED STATES PATENT AND TRADEMARK OFFICE

Commissioner for Patents
United States Patent and Trademark Office
P.O. Box 1450
Alexandria, VA 22313-1450
www.uspto.gov

10646289

BEFORE THE BOARD OF PATENT APPEALS AND INTERFERENCES

Application Number: 10/646,289
Filing Date: August 21, 2003
Appellant(s): HO ET AL.

Michael D. Wiggins
For Appellant

EXAMINER'S ANSWER

This is in response to the appeal brief filed March 06, 2008 appealing from the Office action mailed May 07, 2007.

(1) Real Party in Interest

A statement identifying by name the real party in interest is contained in the brief.

(2) Related Appeals and Interferences

The following are the related appeals, interferences, and judicial proceedings known to the examiner which may be related to, directly affect or be directly affected by or have a bearing on the Board's decision in the pending appeal:

Patent Application No. 10/626,507

(3) Status of Claims

The statement of the status of claims contained in the brief is correct.

(4) Status of Amendments After Final

The appellant's statement of the status of amendments after final rejection contained in the brief is correct.

(5) Summary of Claimed Subject Matter

The summary of claimed subject matter contained in the brief is correct.

(6) Grounds of Rejection to be Reviewed on Appeal

The appellant's statement of the grounds of rejection to be reviewed on appeal is correct.

(7) Claims Appendix

The copy of the appealed claims contained in the Appendix to the brief is correct.

(8) Evidence Relied Upon

6,601,126	Zaidi et al.	7-2003
7,133,972	Jeddeloh	11-2006
2005/0021916	Loafman	1-2005
6,725,334	Barroso et al.	4-2004

(9) Grounds of Rejection

The following ground(s) of rejection are applicable to the appealed claims:

Claim Rejections - 35 USC § 103

1. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

2. Claims 1-2, 5-7, 10, 13-15, 18-19 and 25 are rejected under 35 U.S.C. 103(a) as being unpatentable over Zaidi et al. (US 6,601,126 B1) (Zaidi herein after), Jeddeloh (US 7,133,972 B2) and Loafman (US 2005/0021916 A1).

As per claims 1 and 18, Zaidi teaches memory storage system that is accessed by a first central processing unit (CPU) (fig. 1, item 100 and item 112, also see figs. 20-23), comprising:

a line cache including a plurality of pages that are accessed by the first CPU (fig. 1, item 126. Also caches are known to store pages accessed by CPU, see background art section of present application); and

a first memory device that stores data that is loaded into said line cache when miss occurs (fig.1, item 108 is connected to processor via cache 126 to CPU 112, column 22, lines 65-67, teaches processor use cache to access data from memory 108 and as per present application's background art section, data are loaded from lower latency memories to cache when miss occurs);

a second memory device (figs. 1-2 and 20-23, items 106, 108, 248, 250 and indicated as flash and SDRAM);

a line cache control system that controls data flow between said line cache, the first CPU, said first memory device and said second memory device (CPU accessing data from lower level storage devices through cache teaches cache control system), and that includes:

a first line cache interface that is associated with the first CPU, that receives a first program read request from the first CPU and that generates a first address from said first program read request (figs. 20-22 shows cache of CPU is connected to memory devices through MAC, which inherently teaches interfaces and CPU accessing information from cache as explained above inherently requires generating address);

a first memory interface that communicates with said first memory device and a second memory interface that communicates with said second memory device (figs. 20-23, flash and SDRAM is connected to MAC);

a switch that selectively connects said line cache to one of said first and second memory interfaces, wherein when said line cache receives said first address, said line cache control system compares said first address to stored addresses in said line cache, returns data associated with said first address if match occurs and loads page of said line cache when miss occurs [caches are known to receive addresses from processors and comparing those addresses to stored addresses and providing data to processor if hit occurs and loading pages from lower higher latency memory in case of cache miss (see background of invention in present application)], (column 23, lines 31-34 and lines 41-45, taught as CPUs supply a request and an address, the address includes both the port, device or memory bank address and the requested memory location address. Referring to figs. 20-22, col. 23 lines 22-29, "switched channel memory controller" allows multiple DMA (and processors) to simultaneously communicate with multiple output channels. Also, col. 23, lines 40-45, suggests that CPU bus can be connected to an external flash memory through one channel and SDRAM through another channel. These statements clearly indicate that there are separate and selective communication interfaces between the devices). (switches providing separate/distinct/exclusive [limitation of claim 18] interfaces are known in the art, because a separate and independent interface avoids bus or memory bank conflict

and hence increases the speed of the system, see Jeddeloh, US 7,133,972, fig. 3, col. 4, lines 30-64,).

Zaidi fails to teach loading n pages from sequential locations from memory. Loafman teaches when miss occurs in memory (cache); system preloads consecutive pages from lower latency storage (Loafman, par. [0026]) into memory. It would have been obvious to one having ordinary skill in the art at the time of the invention to modify the memory storage system of Zaidi by prefetching some extra sequential pages when page fault (cache miss) occurs and the requested page is being loaded from higher latency storage to memory (cache), because during sequential access of data it is highly likely that nearby data will be accessed in near future, and by prefetching adjacent (extra) pages will avoid cache miss next time CPU references the sequential data and thus, improving the performance (see Loafman paragraphs [0011]-[0013]).

Zaidi and Loafman fail to teach loading (prefetching) n pages of cache line with the first cache miss (before second miss occurs) as required by the claim. Loafman however teaches that applications/programs read data either sequentially or randomly (also admitted by applicant, remarks submitted March 05, 2007, page 18) and if data being read randomly, prefetching may not work. He further teaches that, *"if data being read randomly and the executing program makes a request to read certain amount of data that resides on two sequential pages and if data is not already in RAM, two page faults will be raised in order to load two pages in the RAM"* (Loafman, par. [0013]). It is apparently clear from above statements that Loafman is not prefetching pages if data being read randomly, but since program requests two sequential pages (even though

program accesses data randomly, it is not totally random and can access pages sequentially), two page faults will be raised (i.e. when program accessing data randomly, no prefetching occurs and hence two page faults required to load two pages), but as mentioned in par. 26, two page faults to two sequential pages, means the virtual memory manager (VMM) infers that the data now being accessed sequentially (par. [0026], *"when program accesses two successive pages (i.e. pages 202 and 204) each using a page fault, the VMM 112 assumes that the program will continue to access data sequentially"*) and hence starts prefetching (admitted by applicant, remarks filed March 05, 2007, page 17, "by observing the pattern used by a program") (Loafman, par. [0013], *"because the pages are sequential, the system may infer that the data being read sequentially; and hence, pre-fetch a block of sequential pages of data"*). Thus, it apparently clear from above explanation, that programs read data either randomly and/or sequentially and if data being read randomly, then pre-fetching creates thrashing of cache (Loafman, pars. [0013], [0027], after reading two sequential pages using two page faults, system infers that now data is being read sequentially, hence starts pre-fetching, but the actual access is still random, so *"the blocks of pages may have been pre-fetched in vain"*), hence Loafman's system confirms that the data being read is read sequentially by having two page faults to two sequential pages before pre-fetching additional data. Thus, it is apparently clear that Loafman uses two page faults (two cache misses) to two sequential pages to confirm that data is being read sequentially. Thus, if the program reads data sequentially, then it would have been obvious to one having ordinary skill in the art at the time of the invention to start prefetching data after

initial (one) miss without waiting for confirmation of whether the data being read sequentially, i.e. before a second miss occurs (as in the system of Loafman). The advantage would be to reduce number of page faults, since it is known that data is being read sequentially, there is no need for confirmation and as taught by Loafman, prefetching works splendidly well in sequential data access and reduces page faults (Loafman, pars. [0010], [0012]), thus reducing the data access latency. Loafman teaches that as long as CPU accesses data in sequential manner, than controller keeps prefetching additional pages (2,4,8 etc. Loafman, fig. 2, items 210, 220, 230) and the next sequential page is already read ahead (pre-fetched) into the cache, hence no additional cache miss occurs.

As per claim 2, Loafman teaches that if program continue sequentially accessing prefetched pages, than prefetching more pages (four and eight and so on) into cache. (Loafman, fig. 2, paragraph [0026]).

Claims 5 and 6 are rejected under same rationales as applied to claims 1 and 2. As Loafman teaches that as long as CPU accesses data in sequential manner, than controller keeps prefetching additional pages (2,4,8 etc. Loafman, fig. 2, items 210, 220, 230) and thus next sequential address is already read ahead in the cache and hence no cache miss occurs.

As per claim 7, Loafman teaches loading 2 pages, then 4 and then 8, sequentially (Loafman, fig. 2, paragraph [0026]), which inherently teaches m^{th} page from n pages (accessing 1^{st} then 2^{nd} and loading 4, and then 8 pages, in case of two pages n = 2 and reading 2^{nd} page teaches reading m^{th} (2^{nd}) page of two pages or 3^{rd} in case of 4

pages preloaded and hence m is greater than one and less than or equal to n and prefetching 4 or 8 pages teaches additional n pages).

Claims 10 and 13-15 are rejected under same rationales as applied to claims 1-2 and 5-7 above.

As per claim 19, combination of Zaidi, Jeddeloh and Loafman teach limitations of independent claim 18, but fail to teach cache translating an address. Computer/storage systems with cache using virtual index (virtual cache) or physical index (physical cache) as well as virtual memory addressing are well known in the art. The physical caches requires address translation for data access from cache and hence cache translating virtual to physical addressing is known in the art, and examiner takes official notice of that. Physical caches are simpler to build; hence it would have been obvious to one having ordinary skill in the art at the time of the invention to use physical cache performing address translation in the system of Zaidi, Jeddeloh and Loafman.

As per claim 25, Loafman teaches loading multiple lines of data from secondary storage device to cache as explained with respect to claim 1 above.

3. Claims 4, 9, 12, 17, 26 and 33 are rejected under **35 U.S.C. 103(a)** as being unpatentable over Zaidi, Jeddeloh and Loafman as applied to claims 1-2, 5-7 and 13-15 above, and further in view of Barroso et al. (US 6,725,334 B2).

As per claims 4, 9, 12 and 17, Zaidi, Jeddeloh and Loafman teach a dual processor system with two caches (fig. 2, items 202 and 214, first and second processors, and items 208 and 224, two caches). Zaidi, Jeddeloh and Loafman

inherently teach cache interface with first and second CPUs and both generates read requests and hence first and second address and providing an exclusive interface as taught in claim 1. Zaidi teaches system with two caches for two processors but fails to teach cache arbitration device which communicates with first and second cache interfaces and resolves cache access conflicts between first and second CPUs. Barroso teaches a second level shared cache with switch (fig. 1, item 130 and 120), which provides interfaces with first and second CPU and arbitrates between first and second CPU (column 4, lines 10-21).

It would have been obvious to one having ordinary skill in the art at the time of invention to modify the multiple cache with multiple processor system of Zaidi and used one cache with switch as taught by Barroso to reduce the cost and the waste of the cache capacity and shared cache avoids duplication of data in individual caches (column 1, lines 45-65).

Claim 26 recites storage system with two processors and ordered data requests. Combination of claims 1 and 4, as taught above teaches dual processor system with shared cache and arbitration device. Jeddeloh teaches switch with arbitration logic to determine memory access priorities and ordering requests according to priority (Jeddeloh, col. 4, line 65 – col. 5, line 2).

As per claim 27, combination of Zaidi, Jeddeloh, Loafman and Barroso teaches limitations of independent claim 26, but fail to teach cache translating an address. Computer/storage systems with cache using virtual index (virtual cache) or physical index (physical cache) as well as virtual memory addressing are well known in the art.

The physical caches requires address translation for data access from cache and hence cache translating virtual to physical addressing is known in the art, and examiner takes official notice of that. Physical caches are simpler to build; hence it would have been obvious to one having ordinary skill in the art at the time of the invention to use physical cache performing address translation in the system of Zaidi, Jeddeloh and Loafman.

As per claim 33, Loafman teaches loading multiple lines of data from secondary storage device to cache as explained with respect to claim 1 above.

4. Claims 20-24 are rejected under **35 U.S.C. 103(a)** as being unpatentable over Zaidi, Jeddeloh and Loafman as applied to claim 18 above, and further in view of Alexander et al. (US 6,131,155).

As per claims 20 and 23, Zaidi, Jeddeloh and Loafman teach all the limitations of independent claim 18, but fail to teach direct interface to bypass cache. Alexander teaches a CPU programmed to bypass a cache when necessary (Alexander, abstract).

It would have been obvious to one having ordinary skill in the art at the time of the invention to utilize the direct access interface to memory, bypassing the cache as taught by Alexander in the system of Zaidi, Jeddeloh and Loafman, because data caches provides performance improvement only if program execution performs repeated accesses of data over a short period of time to a small group of data and large amounts of data transfers degrades the performance by the phenomenon known as threshing, so bypassing a cache and directly reading burst data from memory increases the performance (Alexander, abstract, col. 2, lines 21-56). Also providing a

direct/exclusive and independent interface avoids bus or memory bank conflict as explained with respect to claims 1, 4, 18 and 26 above.

As per limitation of claims 21-22 and 24, Alexander teaches that when necessary in order to fetch or store (reading from and writing/programming to memory device) data items directly from/to the memory, ensuring data accesses exhibiting a high degree of locality are made to the cache, while those accesses that are non-local are made directly to the main memory, bypassing the cache (Alexander, abstract), proves that depending upon addresses certain access are only performed through the direct memory interface, bypassing the cache.

5. Claims 28-32 are rejected under **35 U.S.C. 103(a)** as being unpatentable over Zaidi, Jeddeloh, Loafman and Barroso as applied to claim 26 above, and further in view of Alexander et al. (US 6,131,155).

As per claim 28, Zaidi, Jeddeloh, Loafman and Barroso teach all the limitations of independent claim 26, but fail to teach direct interface to bypass cache. Alexander teaches a CPU programmed to bypass a cache when necessary (Alexander, abstract).

It would have been obvious to one having ordinary skill in the art at the time of the invention to utilize the direct access interface to memory, bypassing the cache as taught by Alexander in the system of Zaidi, Jeddeloh, Loafman and Barroso, because data caches provides performance improvement only if program execution performs repeated accesses of data over a short period of time to a small group of data and large amounts of data transfers degrades the performance by the phenomenon known as

threshing, so bypassing a cache and directly reading burst data from memory increases the performance (Alexander, abstract, col. 2, lines 21-56). Also providing a direct/exclusive and independent interface avoids bus or memory bank conflict as explained with respect to claims 1, 4, 18 and 26 above.

As per limitation of claims 29 and 30-32, Alexander teaches that when necessary in order to fetch or store (reading from and writing/programming to memory device) data items directly from/to the memory, ensuring data accesses exhibiting a high degree of locality are made to the cache, while those accesses that are non-local are made directly to the main memory, bypassing the cache (Alexander, abstract), proves that depending upon addresses certain access are only performed through the direct memory interface, bypassing the cache.

As per limitation of claim 31, Zaidi and Barroso teach an arbiter and MAC (Zaidi, fig. 2, items 242, 244) and switch (Barroso, fig.1, item 120) to resolve memory access conflict (Zaidi, column 23, lines 40-45) but fail to teach arbiter for direct read/write interface. It would have been obvious to one having ordinary skill in the art at the time of the invention would provide arbiter for direct (bypassing cache interface) interface, because when multiple CPUs accessing memory device, providing arbitration avoids the conflict for same data.

(10) Response to Argument

Appellant mainly argues one point, none of the prior art reference teach or suggests pre-fetching/loading n pages of the data before second miss occurs. The

Examiner agrees with Appellant's argument that none of the references explicitly teach pre-fetching/loading n pages of the data on a single miss (i.e. before second miss occurs). However, the Examiner would like to point out that data pre-fetching or read-ahead is a very well known technique used in the art to reduce the number of page faults or cache misses (e.g. see Loafman, par. [0010]):

"Consequently, **to reduce the number of page faults that may occur during the execution of a program, a method known as read-ahead or data pre-fetching is used**. As the name suggests, data pre-fetching involves obtaining data before it is needed."

Loafman further teaches that technique known as spatial data pre-fetching (i.e. fetching the data block/cache line adjacent to the data block/cache line currently being accessed, see par. [0011]) works splendidly when data is being read sequentially (see par. [0012]). The Examiner would like to point out that according to Loafman, the executing applications can read/access data either randomly or sequentially. If the application reads/accesses data randomly than pre-fetching of data will waste resources (e.g. additional time required to load data and space required to store pre-fetched data) (see par. [0013]). The Examiner also contends that very simple spatial pre-fetching technique known in the art pre-fetches an additional line each time a cache miss occurs, i.e. pre-fetching an additional line of data on first cache miss. Here, it is readily understood that **Loafman requires two page faults (or cache miss) to determine that program reads data sequentially, which is also admitted by the Appellant** (see appeal brief, page 11, lines 3-13). However, the Examiner respectfully disagrees with the Appellant's assertion that Loafman requires two page faults regardless of

whether data is being read randomly or sequentially (see appeal brief, page 12, lines 1-4). The Examiner again asserts that according to Loafman applications executing on processors can read/access data either in random pattern or in sequential pattern. Now suppose the application is accessing data randomly and during that random access pattern, if the application requests two successive pages of data using two page faults, than the system will now infer that the application has started accessing data sequentially and thus starts pre-fetching data. However, pre-fetching is disadvantageous when application is accessing data randomly (see Loafman, par. [0013]):

"However, if data is being **read randomly**, spatial **data pre-fetching may not work as well**. For example, suppose an executing program is randomly reading data. Suppose further that the executing program makes a request to read a certain amount of data that resides on two sequential pages."

(From above lines it is clear that the program/application is reading data randomly, however it makes a request to read data from two sequential pages).

Loafman further teaches:

"If the data is not already in RAM, two page faults will be raised in order to load the two pages in the RAM. **Because the pages are sequential, the system may infer that data is being read sequentially; and hence, pre-fetch a block of sequential pages of data.**"

(Since two page faults are raised to load two sequential pages, the system determines that now data is being read sequentially, as such it starts pre-fetching data).

"Since data is being read randomly, it is highly unlikely that future needed data will be on the pre-fetched block of pages. Thus, the block of pages may have been pre-fetched in vain and the physical pages onto which they are placed wasted. As will be explained later, continually pre-fetching unneeded pages of data may place an undue pressure on RAM space."

Thus, it clear from above explanation that Loafman suggests pre-fetching data only when the program/application is reading data sequentially. However, as noted above programs/applications can read/access data either randomly and/or sequentially. Again as noted above if pre-fetching is performed during random access than pre-fetching will not be useful, as such Loafman requires two page faults to two sequential pages to determine whether the program is reading data sequentially and accordingly to start pre-fetching additional data (also admitted by the Appellant, see appeal brief, page 11, lines 3-13, page 13, lines 1-2).

Now, the Examiner would like to point out that claims must be given broadest reasonable interpretation in light of Appellant's disclosure. However, the appellant's claims recite: (claim 1) "wherein when said miss occurs and before second miss occurs, n pages of said line cache are loaded from sequential locations", (claims 5, 26) "wherein after an initial miss, said line cache prevents any additional misses as long as the first CPU addresses sequential locations of said first memory device". Here, it is entirely clear that CPU accesses data sequentially, and person having ordinary skill in the art at the time of the invention, with teaching of Loafman that "pre-fetching works splendidly when data is being read sequentially" in front of him, with a knowledge that the application reads data sequentially, obviously be motivated to start pre-fetching on initial cache miss, because the data pre-fetching technique is mainly used to reduce page faults. It is further noted that Appellant's disclosure suggests reading data sequentially and there is no mention of random reading, accordingly it is inferred that

the claimed invention is directed towards sequential data access and person having ordinary skill in the art would readily determine that there is no need to confirm that if program reads data sequentially by using two page faults.

The Appellant further argues that "the Examiner fails to provide any reference that teaches or suggests any knowledge of sequential data reading that is not based on at least two cache misses" (appeal brief, page 13). The Examiner would like to point out that Loafman teaches system with random and sequential data accesses, and as such he requires two page faults to determine whether the system accesses data sequentially or randomly, however no where in entire disclosure the Appellant elaborates any type of random data access, as such person having ordinary skill in the art may readily infer that the Appellant's system mainly accesses data sequentially and there is no need to confirm sequential access using two page faults.

The dependent claims 2, 4, 6-7, 9, 12, 14-15, 17 and 19-25 as well as separately argued independent claim 26 also remains rejected based on same arguments stated above.

(11) Related Proceeding(s) Appendix

No decision rendered by a court or the Board is identified by the examiner in the Related Appeals and Interferences section of this examiner's answer.

For the above reasons, it is believed that the rejections should be sustained.

Respectfully submitted,

Kaushik Patel

/Kaushik Patel/

Examiner, AU 2188

Conferees:

/Hyung S SOUGH/

Supervisory Patent Examiner, Art Unit 2188

04/23/08

/Manorama Padmanabhan/

WQAS, TC 2100, WG2180